```
{ // 判断被减数和减数的大小
    f = true;
    swap(s1,s2);
    swap(len1,len2);
} // 若减数更大就交换被减数和减数,同时在输出时先输出一个负号
for (int i = 1; i <= len1; i++)
    num1[i] = s1[len1 - i + 1] - '0'; // 按数位拆分后逆序存储
for (int i = 1; i <= len2; i++)
    num2[i] = s2[len2 - i + 1] - '0';
for (int i = 1; i <= len1; i++)
{
    res[i] += num1[i] - num2[i];
    if (res[i] < 0)
    { // 处理借位
        res[i + 1]--;
        res[i] += 10;
    }
}
len3 = len1; // 计算高精度数的长度
while (len3 > 1 && res[len3] == 0)
    len3--; // 删除前导 0
if (f == true)
    putchar('-');
for (int i = len3; i >= 1; i--)
    printf("%d",res[i]); // 从高位到低位依次输出

// 重载高精度减法
bignum operator-(bignum a,bignum b)
{
    bignum res;
    if (a < b)
    {
        res.flag = 1;
        swap(a,b);
    }
    res.len = a.len;
    for (int i = 1; i <= res.len; i++)
    {
        res[i] += a[i] - b[i];
        if (res[i] < 0)
        {
            res[i + 1]--;
            res[i] += 10;
        }
    }
```

```
    while (res.len > 1 && res[res.len] == 0)
        res.len--;
    return res;
}
// 高精度数的减法计算
bignum a,b,c;
a.input();
b.input();
c = a - b;
c.print();
```

参考词条

高精度加法

（陈奕哲　谢秋锋）

1.4.4.3　高精度乘法

高精度乘法是指实现高精度数乘法操作的算法，算法的基本思路是将高精度数按数位拆分后分别存储，再通过模拟乘法竖式来计算结果。高精度乘法的主要步骤如下：

（1）以字符串形式读入两个高精度数，分别表示两个乘数；

（2）将两个高精度数按数位拆分后，逆序存储在两个数组中；

（3）模拟乘法竖式，枚举两个乘数的每一位分别相乘，将结果统计到积的对应位上；

（4）从低位到高位依次处理进位的情况；

（5）从高位到低位依次输出。

代码示例

高精度乘法的核心代码如下。

```
scanf("%s%s",s1 + 1,s2 + 1); // 读入两个高精度数
len1 = strlen(s1 + 1);
len2 = strlen(s2 + 1);
len3 = len1 + len2; // 计算高精度数的长度
for (int i = 1; i <= len1; i++)
    num1[i] = s1[len1 + 1 - i] - '0'; // 按数位拆分后逆序存储
for (int i = 1; i <= len2; i++)
    num2[i] = s2[len2 + 1 - i] - '0';
// 第 i 位和第 j 位的位权分别是 10^(i-1) 和 10^(j-1)
// 故一个乘数的第 i 位与另一乘数的第 j 位乘积的位权是 10^(i+j-2)
// 故累加至积的第 i+j-1 位
for (int i = 1; i <= len1; i++)
    for (int j = 1; j <= len2; j++)
```

```
            res[i + j - 1] += num1[i] * num2[j];
    for (int i = 1; i <= len3; i++)
    {
        if (res[i] >= 10)
        { // 处理进位
            res[i + 1] += res[i] / 10;
            res[i] %= 10;
        }
    }
    while (len3 > 1 && res[len3] == 0)
        len3--; // 删除前导 0
    for (int i = len3; i >= 1; i--)
        printf("%d",res[i]); // 从高位到低位依次输出
    // 重载高精度乘法
    bignum operator* (bignum a,bignum b)
    {
        bignum res;
        res.len = a.len + b.len;
        for (int i = 1; i <= a.len; i++)
            for (int j = 1; j <= b.len; j++)
                res[i + j - 1] += a[i] * b[j];
        for (int i = 1; i <= res.len; i++)
        {
            if (res[i] >= 10)
            {
                res[i + 1] += res[i] / 10;
                res[i] %= 10;
            }
        }
        while (res.len > 1 && res[res.len] == 0)
            res.len--;
        return res;
    }
    // 高精度数的乘法计算
    bignum a,b,c;
    a.input();
    b.input();
    c = a * b;
    c.print();
```

参考词条

高精度乘法

典型题目

1. NOIP1998 普及组 阶乘之和
2. NOIP2000 提高组 乘积最大
3. NOIP2012 提高组 国王游戏

（陈奕哲　谢秋锋）

1.4.4.4　高精度整数除以单精度整数的商和余数

高精度除法是指实现高精度数除法操作的算法。当被除数是高精度数，除数是单精度数时，可以通过模拟除法竖式来计算结果。高精度数除以单精度数算法的主要步骤如下：

（1）读入一个高精度数和一个单精度数，分别表示被除数和除数；

（2）将高精度数按位拆分后，逆序存储在数组中，同时定义一个初始值为 0 的变量 k 表示余数；

（3）模拟除法竖式，从高位到低位依次计算出商的每一位，并不断更新 k；

（4）从高位向低位依次输出数组的值，数组的值为商；输出 k 的值，k 的值为余数。

代码示例

高精度数除以单精度数算法的核心代码如下。

```
scanf("%s%d",s + 1,&n); // 读入一个高精度数和一个单精度数
len1 = len2 = strlen(s + 1); // 计算高精度数的长度
for (int i = 1; i <= len1; i++)
    a[i] = s[len1 - i + 1] - '0'; // 按数位拆分后逆序存储
for (int i = len1; i > 0; i--)
{
    b[i] = (a[i] + k * 10) / n; // 计算商每一位的值
    k = (a[i] + k * 10) %n; // 更新余数的值
}
while (len2 > 1 && b[len2] == 0)
    len2--; // 删除前导 0
for (int i = len2; i >= 1; i--)
    printf("%d",b[i]); // 从高位到低位依次输出商的每一位
printf("\n%d",k); // 输出余数

// 重载高精度除法
bignum operator/(bignum a,int b)
{
    bignum res;
    res.len = a.len;
    long long k = 0;
    for (int i = res.len; i >= 1; i--)
    {
```

```
            res[i] = (a[i] + k * 10) / b;
            k = (a[i] + k * 10) %b;
        }
        while (res.len > 1 && res[res.len] == 0)
            res.len--;
        return res;
    }
    // 重载高精度模运算
    int operator% (bignum a,int b)
    {
        long long k = 0;
        for (int i = a.len; i >= 1; i--)
            k = (a[i] + k * 10) %b;
        return k;
    }
    // 高精度数除以单精度数,求商和余数
    int b,k;
    bignum a,c;
    a.input();
    scanf("%d",&b);
    c = a / b;
    c.print();
    k = a %b;
    printf("\n%d",k);
```

参考词条

高精度除法

（陈奕哲　谢秋锋）

1.4.5　排序算法

1.4.5.1　排序的基本概念

排序就是把一组记录（元素）按照某个域值的递增或递减的次序重新排列的过程。通常将用于排序的域称为排序域或排序项，其值称为排序码。在解决问题时，有序序列常常比无序序列更容易操作，解决问题的效率更高。如在有序序列上进行二分法查找会优于在无序序列上按顺序查找。

目前常见的排序算法有冒泡排序、选择排序、插入排序、归并排序、快速排序、堆排序、桶排序、计数排序、基数排序等。

如果某种排序方法只通过对任意两个元素的比较进行排序，则该方法称为基于比较的排序（comparison sort），或简称为比较排序。常见的比较排序方法包括冒泡排序、选